Code Review

1. **Poorly structured code**
    a. **Smell**: Mismatched responsibilities with Trap class originally checking if the score dropped below 0 (Trap should just be concerned with the trap being touched, and the Score class should be concerned about what the score value means)
    b. **Solution**: moved the score-checking code from Trap class's trapCatch() method to Score class's increaseScore() method
    c. **Relevant code**: the `if (score < 0) {. . .}` has been added to Line 30 of the Score class and removed from the Trap class

2. **Bad/confusing variable names**
    a. **Smell**: variable name in GameManger.java file "widw" and "widh" is not clear to understand
    b. **Solution**: Change it to windowWidth and windowHeight
    c. **Relevant code**: `public int widw` and `public int widh` changed to `public int windowWidth` and `public int widHeight` (Lines 39 and 40 in GameManger, as well as classes that used original variables)

3. **Dead code / unnecessary comments**
    a. **Smell**: unused line of code in the Score class that was initially used for testing.
    b. **Solution**: Get rid of the code for cleaner code.
    c. **Relevant code**: *`// System.out.println(minutes + ":" + gm.seconds);`* -> Got rid of it (originally line 57 in the Score class)

4. **Classes that are too large and/or try to do too much**
    a. **Smell**: the thread run() method in the GameManager class has many lines of code just to create the walls of the maze in our game
    b. **Solution**: create a new class (extract class) called Maze and move the wall-creation code from GameManager to a method in the new class
    c. **Relevant code**: the Maze class (new file) and its instantiation/call in the GameManager class (Lines 61 and 168)

5. **Code duplication**
    a. **Smell**: Hardcoded square size (same raw value used in different files; ex. x += 50 && y += 50 instead of x += variable && y += variable)
    b. **Solution**: put the value in a variable (`public int squareSize` in GameManager class), so if we need to change the square size, we can just change 1 thing in 1 file, without going through multiple files looking for all instances of the original value
    c. **Relevant code**: Line 41 of the GameManager class (`public int squareSize = 50;`) (50 is the original value used), as well as classes that used the value before (Pathfinder, etc.)

6. **Badly structured project**
    a. **Smell**: structure of the project had room for improvement (all files were together in 1 package)
    b. **Solution**: Added a “Frames” package to store all the files related to frames for easier navigation and understanding of file structure
    c. **Relevant code**: Added Frames package to the project and put all the frames files inside it (on Gitlab: src/main/java/group3/demonGame/Frames)

7. **Dead code**
    a. **Smell**: decreaseScore() method of the Score class was never used
    b. **Solution**: get rid of the decreaseScore() method
    c. **Relevant code**: remove the lines of code for decreaseScore() method and its corresponding Javadocs for cleaner code

8. **Misleading variable name**
    a. **Smell**: variable name in Score.java file “increaseScore” does not represent its feature correctly (it will decrease the score if a negative value is passed as a parameter)
    b. **Solution**: Change its name from “increaseScore” to “changeScore”
    c. **Relevant code**: `public void increaseScore -> public void changeScore` (Line 36 in Score class)

9. **Dead code / large class**
    a. **Smell**: The Pathfinder class is large and still contains unused lines of codes in getPath(x, y) that were initially used to implement the method.
    b. **Solution**: Delete these lines of codes to clean up the Pathfinder class.
    c. **Relevant code**: originally Lines 155-168 in the Pathfinder class (`for (int i = 0; i < path.size(); i++)` to `System.out.println(" ");`)

10. **Lack of documentation**
    a. **Smell:** The getPath() of the Pathfinder class method lacks documentation and can be confusing because of all the necessary math and if/else-checks.
    b. **Solution:** Add some comments to make the class easier to understand and modify if needed.
    c. **Relevant code**: starts at Line 128 in the Pathfinder class (the Javadocs)